COMPSCI 690T Coding Theory and Applications Apr 19, 2017

Lecture 21

Instructor: Arya Mazumdar *Scribe: Names Redacted*

1 Covering Codes and Lossy Data Compression

Last time, we started discussing the following lossy data compression problem: given $x \in \{0,1\}^n$, we wish to compress x to $y \in \{0,1\}^k$, then decompress y into $\hat{x} \in \{0,1\}^n$ such that $d(x,\hat{x}) \le \delta n$. Thus we allow for a loss of up to a δ fraction of the information bits, in exchange for compressing the n bit string to a k bit string.

To solve this problem, we introduced the notion of a δn-covering code, defined as a set $M \subseteq \{0,1\}^n$ with the property that for any $x \in \{0,1\}^n$, there exists $c \in M$ such that $d(c,x) \le \delta n$. Thus every vector in $\{0,1\}^n$ is contained in (covered by) some ball of radius δn centered at a codeword of M.

Given a δn-covering code M, if $k = \lceil \log |M| \rceil$ we can achieve data compression of a string of length n down to a string of length k by mapping every vector $x \in \{0,1\}^n$ to a length k representative of the codeword c that covers it; there are only $|M| \le 2^k$ codewords of M, so this is always possible. For example, we could take the representative of c to be the input vector in $\{0,1\}^k$ that encodes to c. If M is not linear, it may be difficult (i. e. require exponential time) to determine what the codeword c that covers x is, but if M is linear, then using any efficient decoding algorithm for linear codes to decode x will decode to the closest codeword $c \in M$, which must satisfy $d(c,x) \le \delta n$.

Last class, we showed that any δn-covering code must satisfy $R = \frac{\log |M|}{n} \le 1 - h(\delta)$, where h is the binary entropy function. Today we will show that this bound is tight, by constructing a covering code achieving this rate. We do this by a random construction: let M consist of vectors chosen randomly and uniformly from $\{0,1\}^n$. We will determine exactly how many vectors should be in M shortly.

If we fix $x \in \{0,1\}^n$ and $c \in M$, then we have

$$\Pr[d(c,x) \le \delta n] = \frac{\sum_{i=0}^{\delta n} \binom{n}{i}}{2^n}.$$

As each $c \in M$ is chosen independently, we then have

$$\Pr[\forall c \in M, d(c,x) > \delta n] = (1 - \frac{\sum_{i=0}^{\delta n} \binom{n}{i}}{2^n})^{|M|}.$$

Then applying a union bound over all 2^n possible choices of x, we have

$$\begin{aligned}\Pr[\exists x \in \{0,1\}^n : \forall c \in M, d(c,x) > \delta n] &\le 2^n (1 - \frac{\sum_{i=0}^{\delta n} \binom{n}{i}}{2^n})^{|M|} \\ &\le e^{-|M|(\frac{\sum_{i=0}^{\delta n} \binom{n}{i}}{2^n}) + n \ln 2}.\end{aligned}$$

Now, if we substitute $|M| = \frac{2^n \cdot 2n \ln 2}{\sum_{i=0}^{\delta n} \binom{n}{i}}$, the probability above is at most 2^{-n}, so with high probability if we choose this many random vectors to form the code, the probability any vector in $\{0,1\}^n$ is not covered is exponentially small. Also, we then have

$$R = \frac{\log |M|}{n} = \frac{n + \log(2n \ln 2) - nh(\delta)}{n} = 1 - h(\delta) + \frac{\log(2n \ln 2)}{n},$$

and the last term goes to 0 for large n, so we achieve a rate of $1 - h(\delta)$ in the limit, meeting the lower bound proved last time.

This construction yields a nonlinear code, so the actual encoding process may not be efficient. It can also be shown that random linear codes are good covering codes, but this is much more difficult. No deterministic constructions of covering codes are known attaining $R = 1 - h(\delta)$.

2 List Decoding Capacity

Recall that a code $\mathcal{C}$ is said to be (pn, L)-list decodable if every Hamming ball of radius pn around a codeword of $\mathcal{C}$ contains at most L codewords.

Here we discuss list decoding capacity, specifically the relationship between the code rate and the error rate that can tolerated using list decoding with a particular list size L. Namely, we will prove that (as long as the code length and list size are sufficiently large) the list decoding capacity is given by $R_L(p) = 1 - h(p)$, where pn is the number of errors that can be tolerated, and $h(p)$ is the (binary) entropy function. More formally, if $n \to \infty$ and L grows polynomially with n, then $R_L(p) = 1 - h(p)$.

Proof Suppose $\mathcal{C}$ is a (pn, L)-list decodable code of size M. Randomly and uniformly pick a point z from $\{0,1\}^n$. For each of the M codewords, we define an indicator X_i for the event that z is in the ball of radius pn around the codeword c_i. In particular, for $c_i \in \mathcal{C}$, we define

$$X_i = \begin{cases} 1, & \text{if } d(c_i, z) \leq pn \\ 0, & \text{otherwise} \end{cases}.$$

We note that the sum of the indicators $X = X_1 + ... + X_M$ is the number of codewords that are in the ball of radius pn around Z, and then consider the expected value of this sum, $E[X]$:

$$\begin{aligned} E[X] &= E[X_1] + E[X_2] + \cdots + E[X_M] \\ &= M \cdot E[X_i] \\ &= M \cdot P(X_i = 1) \\ &= M \cdot P(d(c_i, z) \leq pn). \end{aligned}$$

Thus $E[X] = M \cdot \frac{\sum_{i=0}^{pn} \binom{n}{i}}{2^n}$. Note that $E[X] = \frac{M\binom{n}{i}}{2^n} \leq L$, because there must exist some point with at least $E[X]$ codewords in a ball of radius pn around it, which implies $M \leq \frac{L \cdot 2^n}{\sum_{i=0}^{pn}\binom{n}{i}}$. This gives us $\frac{\log M}{n} \leq 1 - h(p) + \frac{\log L}{n}$, and thus $R_L(p) \leq 1 - h(p)$, as when L grows polynomially in n, $\frac{\log L}{n}$ goes to 0 for large n.

Now we show a construction which achieves this capacity. To do so, we let the code $\mathcal{C}$ be formed by independently and uniformly picking M points from $\{0,1\}^n$ at random. We define the bad event $E_{x,c_1,c_2,\ldots,c_{L+1}}$ to be that the $L+1$ codewords $c_1, \ldots, c_{L+1}$ all lie in the ball of radius pn around x. For a particular choice of x and codewords $c_1, \ldots, c_{L+1}$, the probability of a bad event is

$$P(E_{x,c_1,\ldots,c_{L+1}}) = \left(\frac{\sum_{i=0}^{pm}\binom{n}{i}}{2^n}\right)^{L+1},$$

so applying a union bound over all possible choices of x and all possible choices of a set of $L+1$ codewords, we have that the probability of any bad event occurring is bounded by

$$\begin{aligned} P(\exists x, c_1, \ldots, c_{L+1} : E_{x,c_1 \ldots c_{L+1}} \text{ occurs}) &\leq 2^n \binom{M}{L+1} \left(\frac{\sum_{i=0}^{pn}\binom{n}{i}}{2^n}\right)^{L+1} \\ &\leq 2^n M^{L+1} 2^{-n(1-h(p))(L+1)} \\ &= 2^{n+(L+1)\log M - n(L+1)(1-h(p))} \\ &= 2^{n(L+1)(\frac{1}{L+1} + \frac{\log M}{n} - (1-h(p)))}. \end{aligned}$$

If we set this probability to be equal to 2^{-n} so that it goes to 0 exponentially with n, we find $R = \frac{\log M}{n} = 1 - h(p) - \frac{2}{L+1}$, which goes to $1 - h(p)$ as L becomes large. ■

3 Introduction to Polar Codes

Next, we will begin studying polar codes, a family of codes which achieve the Shannon capacity $1 - h(p)$ in the random error model where codewords are transmitted one bit at a time, and each bit is flipped with probability p. We will first use some of the information-theoretic tools we developed to analyze the wiretap channel to get some intuition for the idea behind polar codes.

To set things up, let x_1 and x_2 be bits drawn from $Ber(p)$, so each is 1 with probability p. Then suppose we have the following circuit which adds x_1 and x_2:

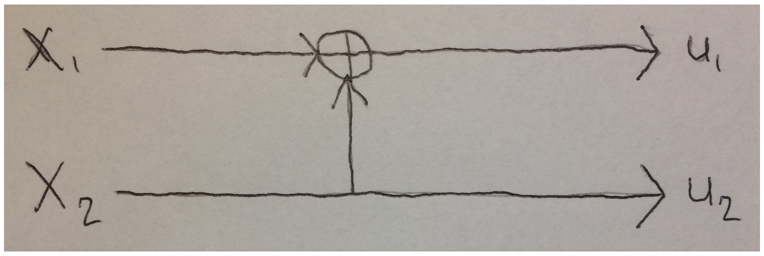

Figure 1: A circuit that adds x_1 and x_2.

As x_1 and x_2 are independent, $H(x_1, x_2) = H(x_1) + H(x_2) = 2H(x_1)$. But also x_1 and x_2 are completely determined by u_1 and u_2, so we must have $H(u_1, u_2) = 2H(x_1)$. Then by the chain rule, $H(u_1, u_2) = H(u_1) + H(u_2|u_1)$. We can compute that $\Pr[u_1 = 1] = 2p(1-p)$, whereas $\Pr[x_1 = 1] = p$, so as H is increasing on the interval $[0, \frac{1}{2}]$, on this interval we have $H(u_1) \geq H(x_1)$, as $2p(1-p) \geq p$, with equality when $p = \frac{1}{2}$. Putting this all together, we have

$$H(u_1) \geq H(x_1) \geq H(u_2|u_1).$$

Thus we have taken two "equally random" bits x_1 and x_2, and using the circuit, created two "unequally random" bits u_1 and u_2.

We can perform a similar process for more than 2 bits, by inductively creating larger and larger circuits. The following circuit performs a similar procedure for 4 bits:

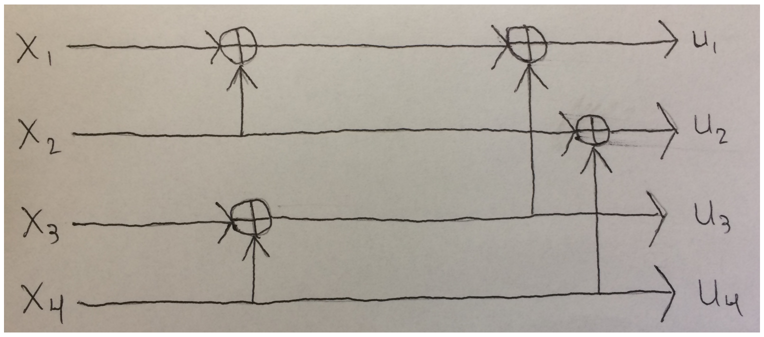

Figure 2: A circuit that adds 4 bits together.

Instead of working with circuits directly, we can view the circuit as matrix multiplication; the two bit adder corresponds to

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

Then if we let $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, the matrix for the 4-bit adder is the block matrix

$$\begin{pmatrix} A & A \\ 0 & A \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

More generally, recall that the Kronecker product of an $m_1 \times n_1$ matrix A and an $m_2 \times n_2$ matrix B, written $A \otimes B$, is the $m_1 m_2 \times n_1 n_2$ block matrix

$$A \otimes B = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{n_1 1} \\ a_{21} & a_{22} & \cdots & a_{n_1 2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1} & a_{m_1 2} & \cdots & a_{m_1 n_1} \end{pmatrix} \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{n_1 1}B \\ a_{21}B & a_{22}B & \cdots & a_{n_1 2}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1}B & a_{m_1 2}B & \cdots & a_{m_1 n_1}B \end{pmatrix}.$$

Then the matrix corresponding to the circuit that adds 2^k bits is the Kronecker product of A with itself k times, written $A^{\otimes k}$. Even though this matrix is extremely large, because it is defined by repeatedly applying this simple rule we can work with it very efficiently. As the matrix becomes larger and larger, the entropy of some u_i will become closer and closer to 1, while for others it becomes closer and closer to 0. This is the reason for the "polar" in "polar" codes; the entropy of the resulting bits is polarized to either 0 or 1. Next time we will see how to use these ideas to actually construct polar codes.